Ejes

Ya con la maestría en el uso de escalas en D3, ahora tenemos el siguiente diagrama de dispersión:

Si añadimos los ejes horizontal y vertical, podemos deshacernos de esos números rojos tan horribles que congestionan el diagrama.

Introducción a Ejes

De manera similar a las funciones de escala, los ejes de D3 son realmente *funciones* a

las que se les pueden definir los parámetros. A diferencia de las escalas, cuando se invoca una función de eje, ésta no devuelve un valor, sino que genera los elementos visuales del eje, incluyendo líneas, etiquetas y marcadores.

Tome nota que las funciones de ejes son específicas a SVG, puesto que generan elementos de SVG. También, el objetivo de los ejes es de que se usen en escalas cuantitativas (y no ordinales).

Construcción del Eje

Use `d3.svg.axis()` para cerrar una función genérica de eje.

```
var xAxis = d3.svg.axis();
```

Como mínimo, cada eje debe saber sobre qué *escala* debe operar. Acá le pasamos `xScale` del código del diagrama de dispersión:

```
xAxis.scale(xScale);
```

También podemos especificar en donde deben aparecer las etiquetas con respecto al eje en sí. Por defecto se despliegan en la parte baja o `bottom`, lo que significa que aparecerán debajo de la línea del eje (Aunque esto se asigna por defecto, no sobra especificarlo explícitamente.)

```
xAxis.orient("bottom");
```

Por supuesto que podemos ser más concisos y encadenar todo esto en una sola línea:

```
var xAxis = d3.svg.axis()
            .scale(xScale)
            .orient("bottom");
```

Por último, para generar el eje e insertar todos estos marcadores pequeños y etiquetas dentro de nuestro SVG, debemos *llamar* a la función `xAxis`. Voy a añadir este código al final de nuestro “script”, de tal manera que el eje se genere después de los demás elementos del SVG:

```
svg.append("g")
    .call(xAxis);
```

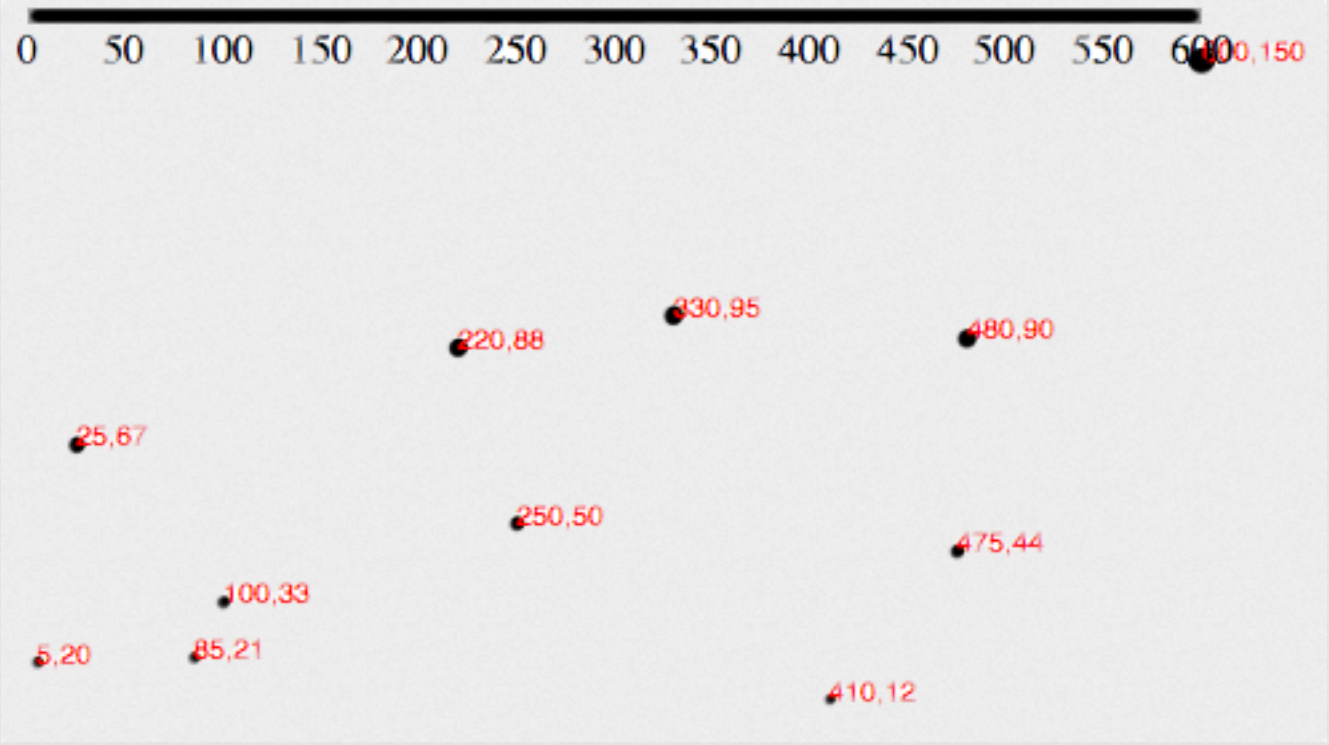
La función `call()` de D3 toma una *selección* como entrada y entrega esa selección a cualquier *función*. Entonces, en este caso, hemos añadido un nuevo elemento de grupo `g` que contiene todos los elementos de los ejes que están para ser generados. (La `g` no es estrictamente necesaria, pero ayuda a mantener organizados a los elementos y nos permite aplicar una clase al grupo entero, que es lo que voy a hacer en un instante.)

La `g` se convierte en la selección para el siguiente eslabón de la cadena. `call()` entrega esta selección a la función `xAxis`, para que se genere nuestro eje dentro del la nueva `g`. Las líneas de código de arriba son la versión limpia y abreviada de su equivalente:

```
svg.append("g")
    .call(d3.svg.axis()
            .scale(xScale)
            .orient("bottom"));
```

Ve, es posible empaquetar toda la función de eje dentro de `call()`, sin embargo es más fácil por lo general para nuestras mentes definir primero la función y luego invocarlas.

En todo caso, acá se ve



Haciendo Limpieza

Técnicamente, ese es un eje, pero no es ni agradable a la vista ni útil. Para limpiarlo, empecemos por asignarle una clase de `axis` a nuestro nuevo elemento `g`, de tal manera que lo podamos usar con CSS:

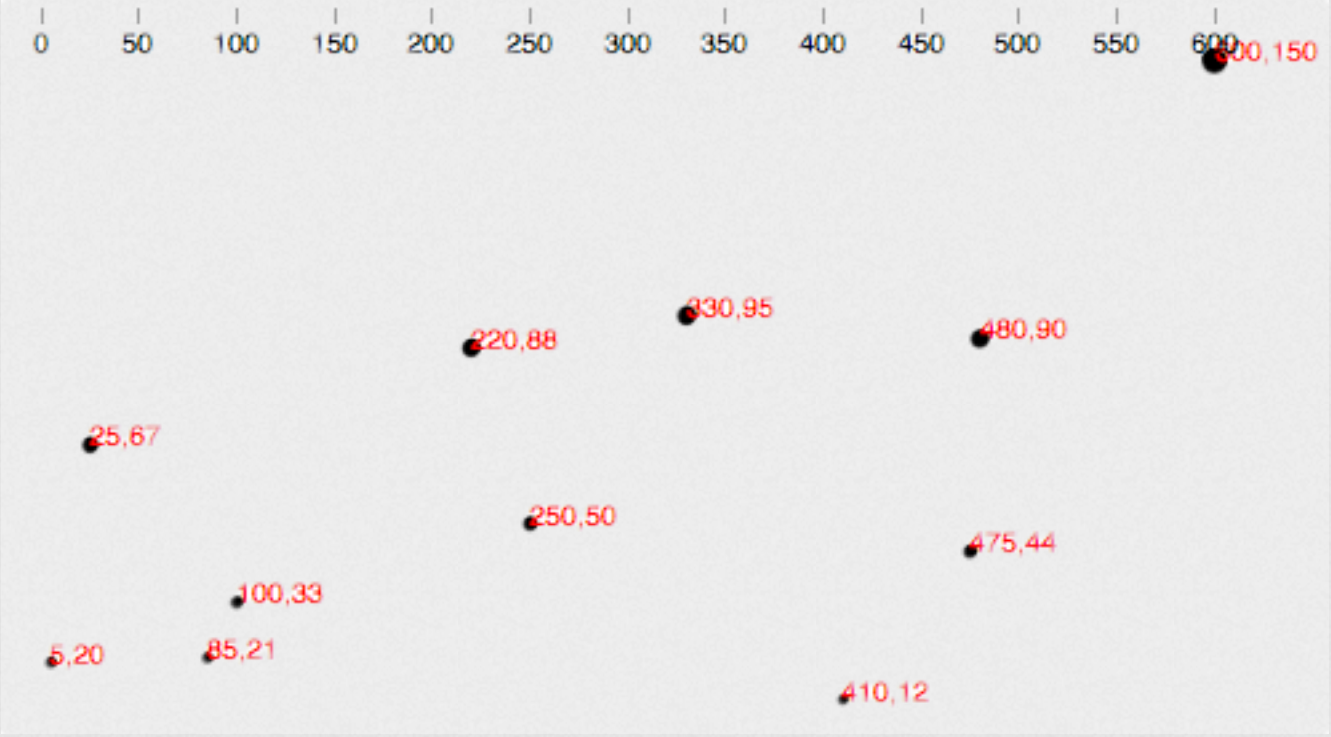
```
svg.append("g")
    .attr("class", "axis")  //Assign "axis" class
    .call(xAxis);
```

Después, introducimos nuestros primeros estilos de CSS, en la etiqueta `<head>` de nuestra página:

```
.axis path,
.axis line {
    fill: none;
    stroke: black;
    shape-rendering: crispEdges;
        }

.axis text {
    font-family: sans-serif;
    font-size: 11px;
}
```

La propiedad `shape-rendering` es un atributo de SVG, que se utiliza para asegurarse de que nuestros ejes y marcadores estén perfectos a nivel de pixel. No podemos tener ejes borrosos!

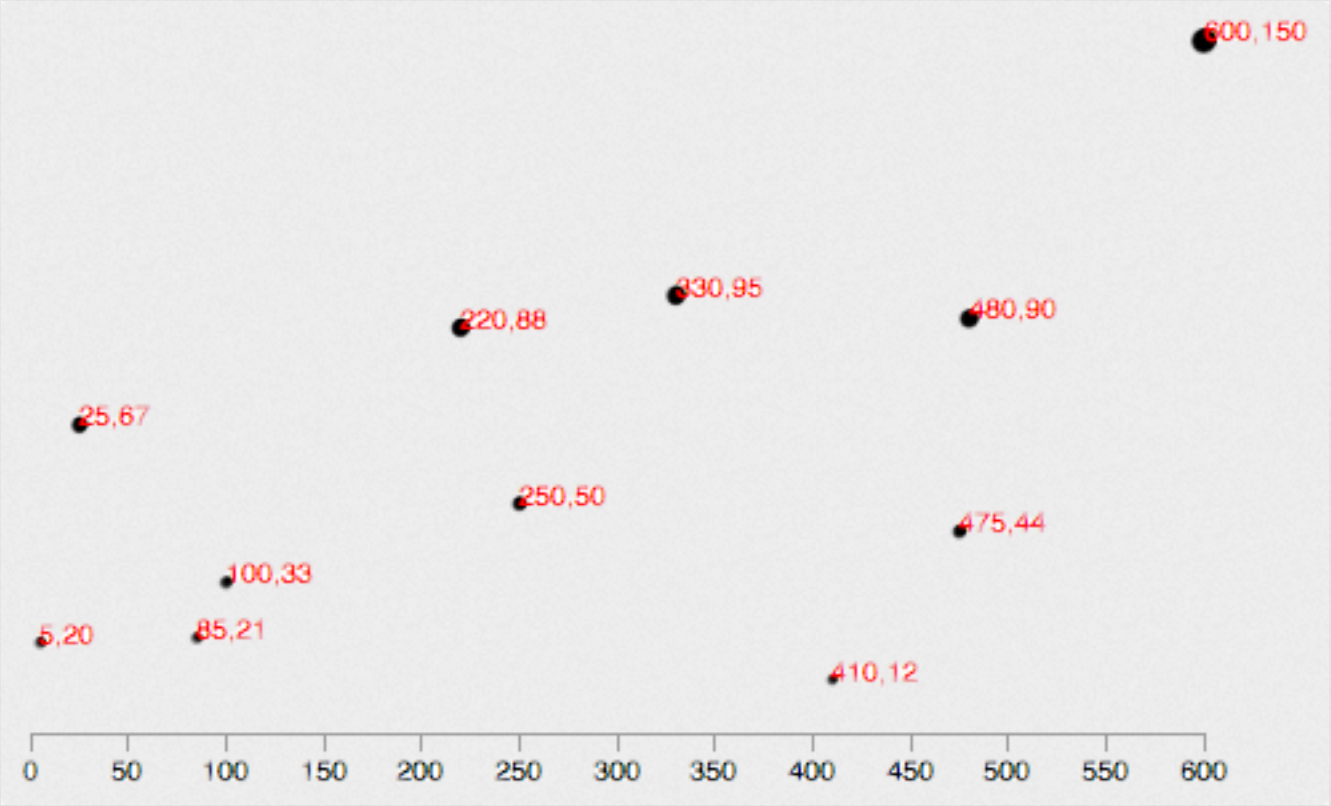


Se ve mejor, pero la parte de arriba del eje está cortada, y queremos que esté en la parte inferior del diagrama. Podemos `transform` (transformar) el grupo entero del eje, empujándolo al fondo:

```
svg.append("g")
    .attr("class", "axis")
    .attr("transform", "translate(0," + (h - padding) + ")")
    .call(xAxis);
```

Tome nota del uso de (`h - padding`), pra que el borde de arriba se le asigne `h`, la altura de

la imagen completa, menos el valor de `padding` (colchón) que creamos anteriormente.



Mucho mejor! Acá está el código que llevamos hasta el momento.

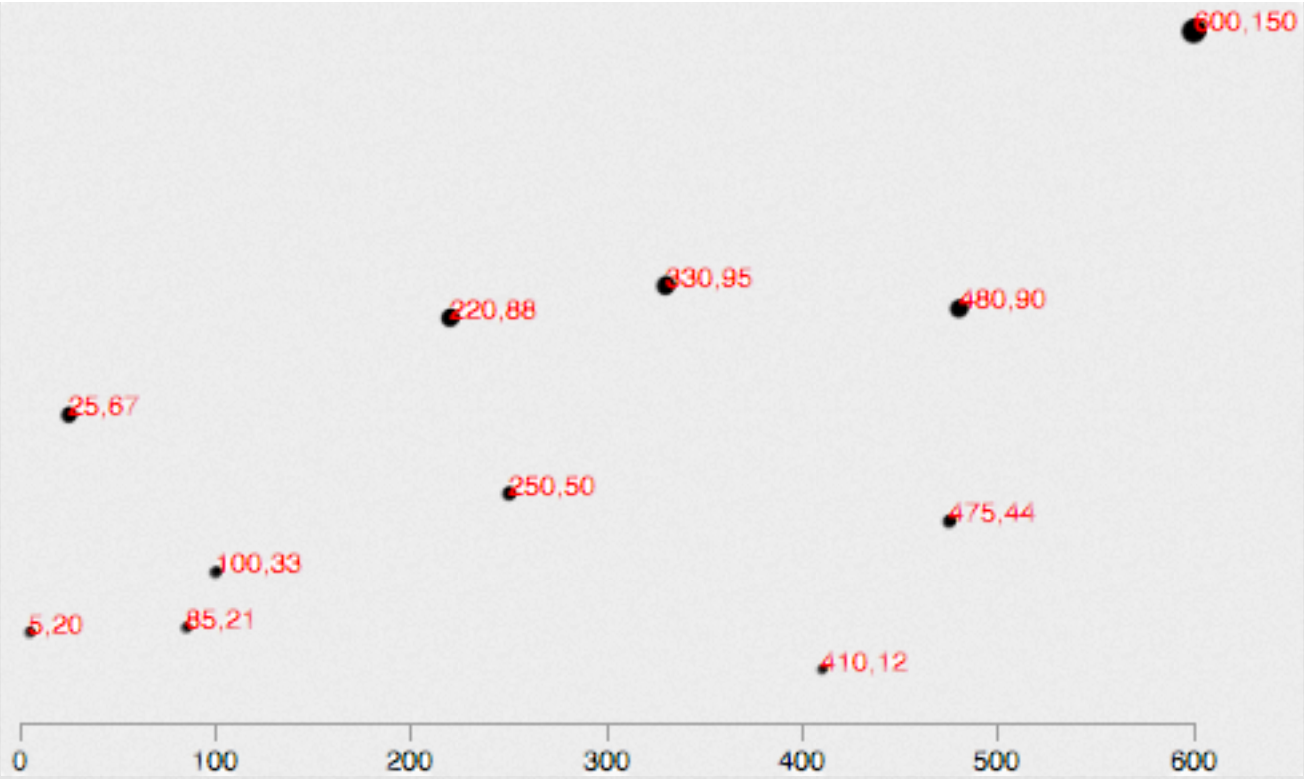
Buscando Marcadores

Los marcadores en D3 proporcionan información. Añadir más marcadores es necesariamente mejor, pero después de cierto punto empiezan a congestionar el diagrama. Podrá notar que nunca hemos especificado cuántos marcadores se deben incluir en el eje y a qué intervalos deben aparecer. Sin instrucciones claras, D3 en forma

automática y mágica examinó nuestra escala `xScale` y determinó a su juicio cuántos marcadores debió incluir y a qué intervalos (cada 50 en este caso).

Como puede imaginárselo, es posible adecuar todos los aspectos de los ejes, empezando con el número aproximado de marcadores, usando la función `ticks()`:

```
var xAxis = d3.svg.axis()
              .scale(xScale)
              .orient("bottom")
              .ticks(5);  //Número aproximado de marcadores
```



Acá está este código

Probablemente notó que cuando aún cuando especificamos solamente cinco marcadores, D3 ha tomado la decisión ejecutiva y pedido un total de siete. Esto es porque D3 nos está protegiendo y calculó que si se incluían solo *cinco* marcadores, esto habría requerido dividir el dominio de entrada en valores no muy vistosos – es este caso 0, 150, 300, 450, y 600. D3 interpreta el valor de `ticks()` solamente como una sugerencia y lo sobreescribe con lo que considere con los valores que considere como los más legibles – en este caso intervalos de 100 – así sea que esto requiera añadir más o disminuir el número de marcadores que usted solicitó. Esto es una funcionalidad

totalmente brillante que incrementa la escalabilidad de su diseño: a medida que cambia su conjunto de datos, y el dominio de entrada se expande o se contrae (números más grandes o más pequeños), D3 se asegura que las etiquetas de los marcadores permanezcan legibles y fáciles de leer.

Y No?

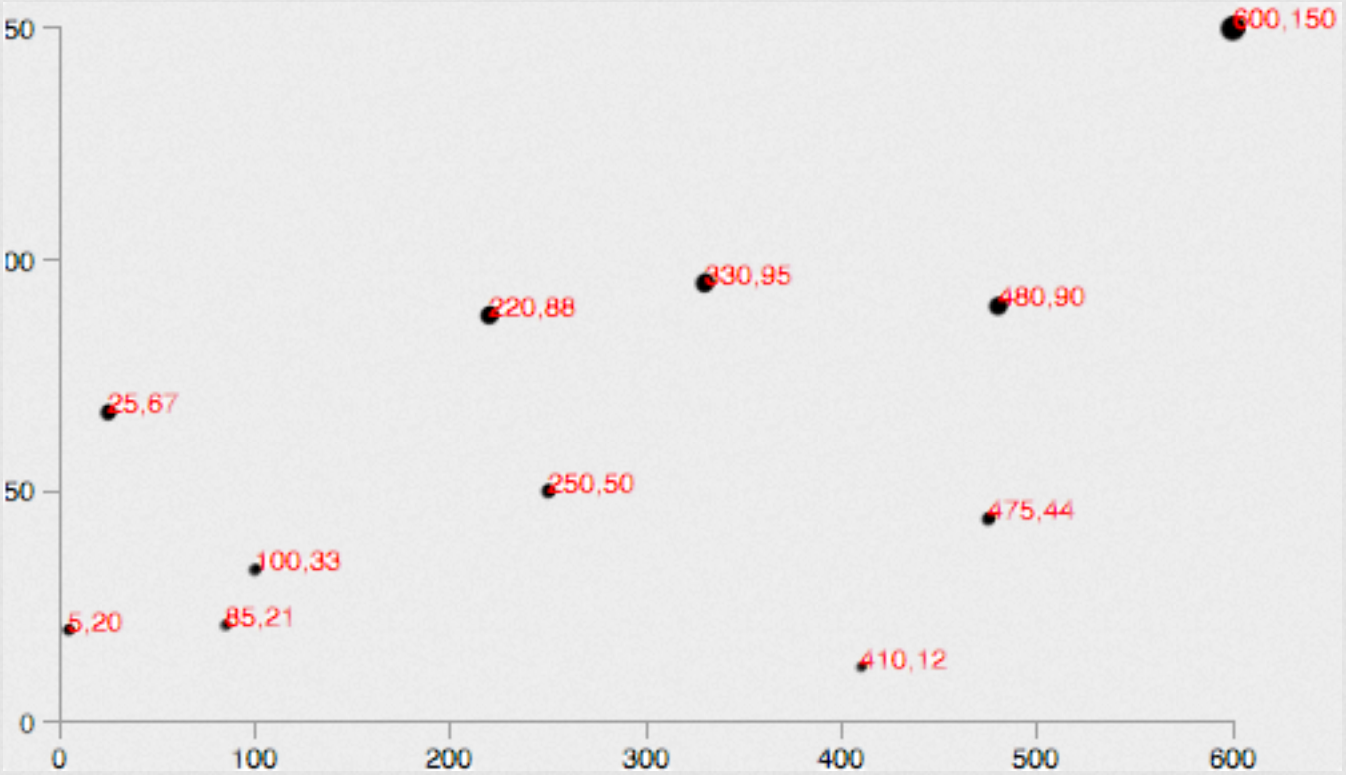
Es hora de añadirle etiquetas al eje vertical! Si copiamos y cambiamos el código que ya escribimos para `xAxis`, debemos añadir esto en la parte de arriba de nuestro código

```
//Define Y axis
var yAxis = d3.svg.axis()
          .scale(yScale)
          .orient("left")
          .ticks(5);
```

y esto hacia la parte de abajo:

```
//Create Y axis
svg.append("g")
    .attr("class", "axis")
    .attr("transform", "translate(" + padding + ",0)")
    .call(yAxis);
```

Puede darse cuenta que las etiquetas están orientadas hacia la izquierda (`left`) y que el `yAxis` perteneciente al grupo `g` se ha trasladado hacia la derecha en la cantidad que se ha determinado con `padding` (colchón).

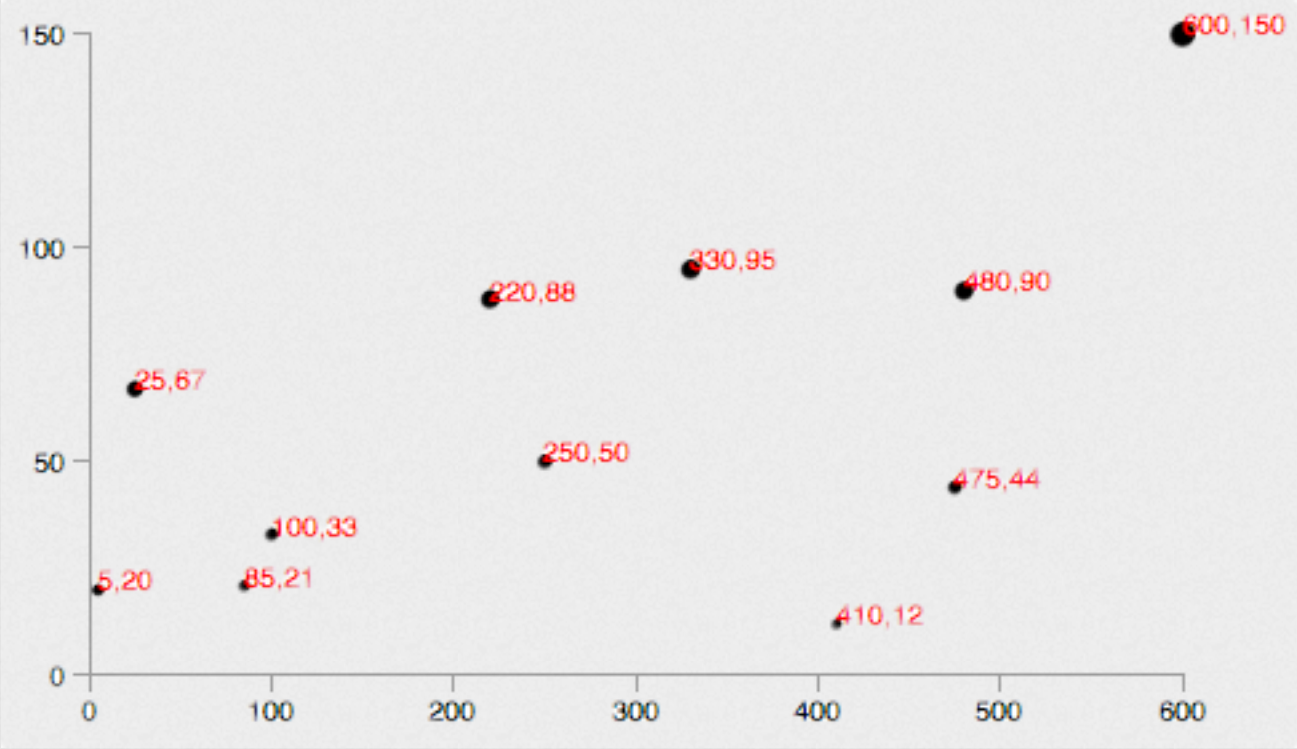


Esto está empezando a verse como algo real! Pero las etiquetas de `yAxis` están quedando cortadas. Para darles más espacio a la izquierda, voy a incrementar el valor de `padding` de 20 a 30:

```
var padding = 30;
```

Por supuesto, también se habría podido introducir variables separadas de `padding` para cada eje, por decir algo `xPadding` y `yPadding` , para obtener aún más control sobre el despliegue.

Acá está el código y así se ve:

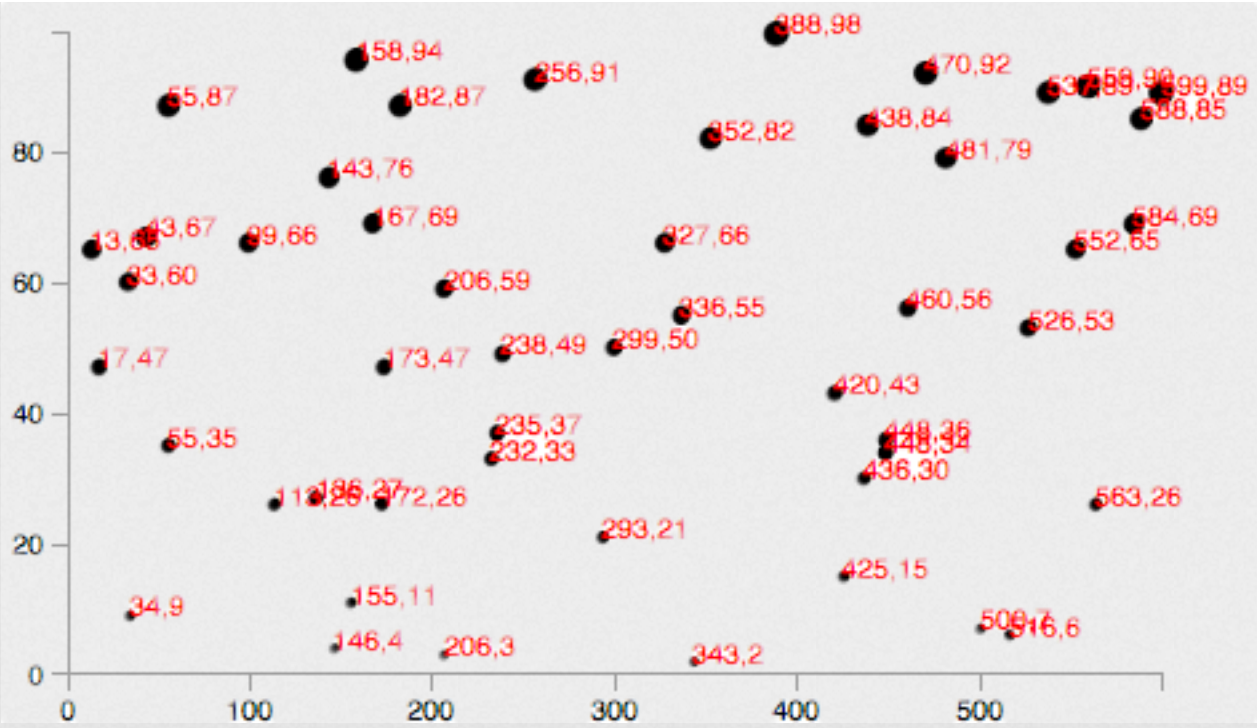


Toques Finales

Para probarle que nuestros nuevos ejes son dinámicos y escalables, quiero reemplazar nuestro conjunto de datos estático por una serie de números generados al azar:

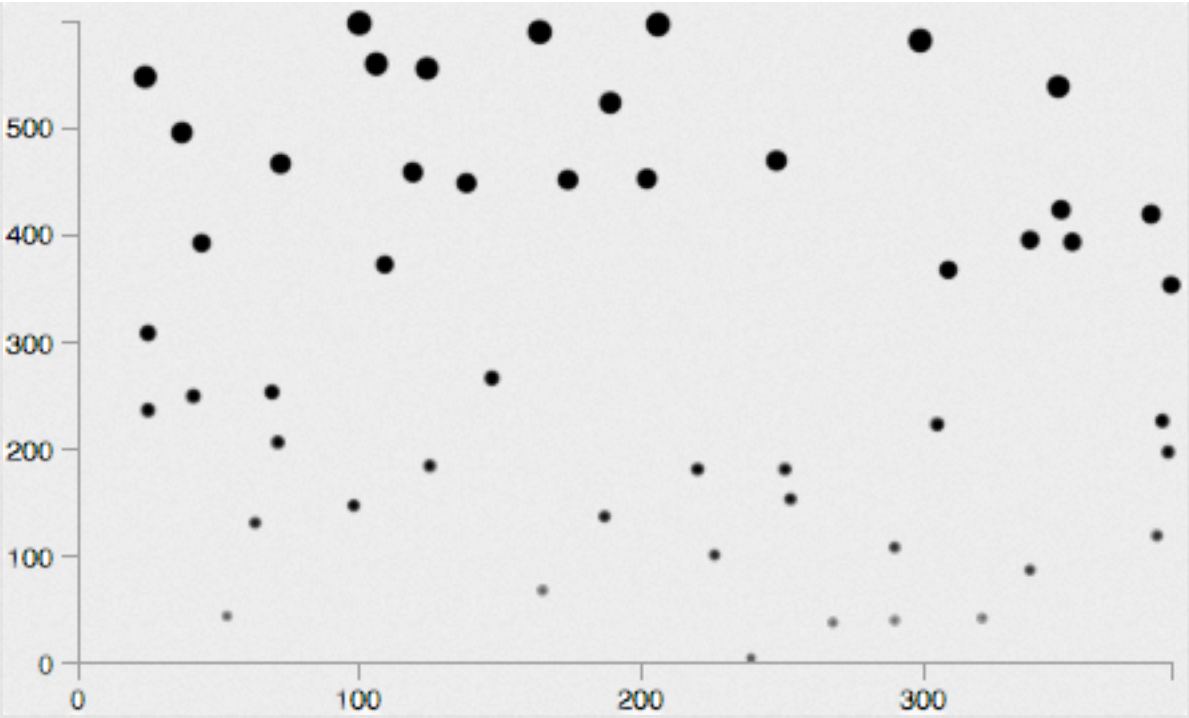
```
//Dynamic, random dataset
var dataset = [];
var numDataPoints = 50;
var xRange = Math.random() * 1000;
var yRange = Math.random() * 1000;
for (var i = 0; i < numDataPoints; i++) {
    var newNumber1 = Math.round(Math.random() * xRange);
    var newNumber2 = Math.round(Math.random() * yRange);
    dataset.push([newNumber1, newNumber2]);
    }
```

Este código inicializa un arreglo vacío, y luego lo recorre 50 veces, selecciona dos números al azar cada vez y añade (“empuja”) ese par de datos al arreglo `dataset`.



Intente el código aquí. Cada vez que refresque la página verá valores diferentes. Mire cómo los dos ejes se ajustan a los nuevos dominios y cómo se escogen marcadores y etiquetas conforme a los cambios.

Ya que he dicho lo que tenía decir al respecto, creo que podemos por fin omitir esas etiquetas rojas tan feas y omitir las líneas de código relevante:



Nuestro código final del diagrama de dispersión!

Formateo de las Etiquetas de Marcadores

Una última cosa: Hasta el momento hemos estado trabajando con números enteros – que son agradables y fáciles. Pero por lo general los datos son más complejos y en esos casos, es posible que quiera tener más control sobre los formatos de las etiquetas de los ejes. Ingrese `tickFormat()`, que permite especificar cómo puede formatear los números. Por ejemplo, si quiere incluir tres puntos decimales o mostrar los valores como porcentajes, o ambos.

En este caso, podría definir una función para formateo de números primero. Esta, por

ejemplo, dice que se deben tratar como porcentajes con un solo número decimal. (Revise el documento de referencia de d3.format() para ver más opciones).

```
var formatAsPercentage = d3.format(".1%");
```

Luego, dígale a su eje que use esta función de formateo para las etiquetas, p. ej.:

```
xAxis.tickFormat(formatAsPercentage);
```

Consejo para desarrolladores: Yo encuentro que lo más fácil es chequear estas funciones de formateo en la consola de JavaScript. Por ejemplo, abra cualquier página que carga D3, como nuestro diagrama de dispersión final y escriba su regla de formateo en la consola. Luego, chequéela entregándole un valor, tal como lo haría con cualquier otra función:

```
> var formatAsPercentage = d3.format(".1%");
  undefined
> formatAsPercentage(0.54321)
  "54.3%"
>
```

Puede rever que el dato `0.54321` se convierte a `54.3%` para efectos de despliegue – perfecto!

Intente el código aquí. El formato de porcentaje no tiene sentido con el conjunto de datos actual, pero como ejercicio, puede intentar modificar cómo es que se generan los

números aleatorios, para lograr valores que no son enteros, pero que sean más apropiados. También puede experimentar con la función de formateo en sí.

Ha terminado el tutorial de D3, pero es posible que añada más en el futuro. Para recibir notificaciones en el futuro, sígame por Twitter o observe este RSS feed

---

Este tutorial fue traducido por Gabriel Coch
Todo el contenido fue desarrollado por y le pertenece a Scott Murray